



Runtime randomization and perturbation for virtual machines.

JAVIER CABRERA ARTEAGA

Licentiate Thesis in [Research Subject - as it is in your ISP]
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden [2022]

TRITA-ICT XXXX:XX
ISBN XXX-XX-XXXX-XXX-X

KTH School of Information and
Communication Technology
SE-164 40 Kista
SWEDEN

Akademisk avhandling som med tillstånd av Kungl Tekniska högskolan framlägg-
es till offentlig granskning för avläggande av licentiatexamen i [ämne/subject]
[veckodag/weekday] den [dag/day] [månad/month] [år/2022] klockan [tid/time] i
[sal/hall], Electrum, Kungl Tekniska högskolan, Kistagången 16, Kista.

© Javier Cabrera Arteaga, [month] [2022]

Tryck: Universitetsservice US AB

Abstract

Write your abstract here...

Keywords: Keyword1, keyword2, ...

Sammanfattning

Write your Swedish summary (popular description) here...

Keywords: Keyword1, keyword2, ...

Acknowledgements

Write your professional acknowledgements here...

Acknowledgements are used to thank all persons who have helped in carrying out the research and to the research organizations/institutions and/or companies for funding the research.

Name Surname,
Place, Date

Contents

Contents	vi
List of Figures	viii
List of Tables	ix
List of Acronyms	xi
1 Introduction	1
1.1 Motivation	1
1.1.1 Why variants ?	1
1.1.2 Research questions	1
1.2 Contributions	1
2 Background and State of the art	3
2.1 CROW	3
3 Methodology	5
3.1 Corpora	5
3.2 RQ1. To what extent can we generate program variants for Web- Assembly?	7
3.3 RQ2. To what extent are the generated variants dynamically different?	9
3.4 RQ3. To what extent can the artificial variants be used to enforce security on Edge-Cloud platforms?	11
3.5 Conclusions	13
4 Results	15
4.1 RQ1. To what extent can we generate program variants for Web- Assembly?	15
4.2 Answer to RQ1.	17
4.3 RQ2. To what extent are the generated variants dynamically different?	17
4.4 Answer to RQ2.	21

<i>CONTENTS</i>	vii
-----------------	-----

4.5 RQ3. To what extent can the artificial variants be used to enforce security on Edge-Cloud platforms?	21
4.6 Answer to RQ3.	25
4.7 Conclusions	26

5 Conclusions	27
----------------------	-----------

Bibliography	29
---------------------	-----------

List of Figures

2.1	CROW workflow to generate program variants. CROW takes C/C++ source codes or LLVM bitcodes to look for code blocks that can be replaced by semantically equivalent code and generates program variants by combining them.	3
3.1	The program variants generation for RQ1.	7
3.2	Population study methodology for each original corpora and the variants generated in RQ1.	9
3.3	Multivariant binary creation and workflow for RQ3.	11
4.1	Pairwise of Metric 2 values in logarithmic scale. Each vertical plot represents a program and its variants. The plot only contains the programs for which we generate more than one variant, <i>i.e.</i> , more than one pair of programs comparisons.	18
4.2	Ratio of Metric 2 with non-zero values for each program's population. Each vertical line represents the number of dt_dyn different from zero in a pairwise comparison of each program and its variants. The plot only contains the 82/239 with at least one pair comparison with zero value.	19
4.3	Execution time distributions for <code>Base64_decode</code> and <code>Hilbert_curve</code> program and their variants in top and bottom figures respectively. Baseline execution time mean is highlighted with the magenta horizontal line.	20
4.4	Ratio of unique execution traces for each endpoint on each edge node. The X axis illustrates the edge nodes. The Y axis annotates the name of the endpoint. In the plot, for a given (x,y) pair, there is blue point representing the Metric 4 value in a set of 100 collected execution traces.	22
4.5	Execution time distributions. Each subplot represents the distribution for a single program, blue for the original program and green for the multivariant binary. The X axis shows the execution time in milliseconds and the Y axis shows the density distribution in logarithmic scale.	25

List of Tables

3.1	Corpora description. The table is composed by the name of the corpus, the number of programs, the number of functions, the lines of code range and the location of the corpus.	7
4.1	General diversification results. The table is composed by the name of the corpus, the number of functions, the number of succesfully diversified functions, the number of non-diversified functions and the cumulative number of variants.	16
4.2	Execution trace diversity over the edge-cloud computing platform. The table is formed of 6 columns: the name of the program, the normalized Shannon Entropy value (Metric 5), the median size of the execution traces, the standard deviation for the trace lengths the number of executed dispatchers (#Diversified) and the number of total random choices taken during all the 6400 executions (#Runtime choices).	24

List of Acronyms

Wasm
DTW

WebAssembly
Dynamic Time Warping

